Classical Simulation of Shor's Algorithm

Kihyo Kwon, Fakhar Zaman, Junaid ur Rehman, and Hyundong Shin
Department of Electronic Engineering, Kyung Hee University, Korea
Email: hshin@khu.ac.kr

***Abstract*—We provide the classical simulation of a compiled version of Shor's algorithm to factor 15. The classical simulation is performed on a simulator designed on Matlab, which is capable of simulating general quantum circuits with different noise models in the circuit. In the noisy simulation, we introduce depolarizing and amplitude damping noise whose strength matches that of actual noise in the IBM quantum device. A comparison of noisy simulation and implementation on IBM quantum devices shows that the simulator closely mimics the behavior of IBM quantum devices by choosing appropriate noise strengths.**

I. Introduction

Shor's algorithm [1] is arguably the most celebrated result in the field of quantum computing. Prior to its proposal in 1994, it was known that quantum computers can provide exponential speedup in some tasks, e.g., Deutch's algorithm. However, such practical speedup was not known for any task of practical relevance. Importance of Shor's algorithm stems from the practical relevance of the task it performs, factoring a composite number in its primes. This problem is suspected but not proven to be NP-complete. Shor's algorithm solves it in polynomial time; offering an exponential speedup over the best known classical algorithm.

Shor's algorithm has direct applications in cryptography. The prime factorization ability of Shor's algorithm has the power to move the RSA algorithm [2], widely used in modern times, to a page in a history book. The stability of RSA cryptosystem is based on the fact that it is difficult to prime factoring large numbers, and Shor's algorithm, which processes prime factoring within the polynomial time, shakes this foundation. Google's quantum computer research team recently claimed that it would take only about 8 hours to prime factoring 2048 bits of natural number using quantum computers [3]. If this big number is really able to prime factoring within 8 hours, then all RSA cryptosystem should be considered completely neutralized. To break existing cryptosystem, a fairly large quantum computer system is required, but in 2001 it succeeded in factoring 15 [4], and in 2012 it succeeded in factoring 21 [5]. However, Shor's algorithm is certainly a major axis of quantum computer development in that the expectation of using quantum algorithms to solve difficult problems much faster than classical computers is the driving force for developing quantum computing.

In this paper, we factor 15 using the compiled version of Shor's algorithm. We simulated two cases using MATLAB App Designer, noiseless case and which is applied gate level noise case. We simulate the noise gates by applying representative noise channels, depolarizing channel and amplitude channel. Also, simulation results are verified through results on the IBM quantum device.

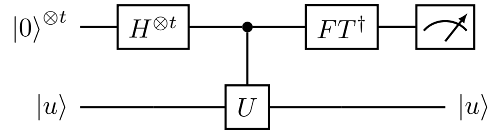


Fig. 1. The circuit of Shor's algorithm implementation.

II. Shor's Algorithm

Shor's algorithm is a quantum algorithm that decomposes a integer number N into prime factors using the time $O\big((\log N)^3\big)$ and the storage space $O(\log N)$. Similar to other quantum algorithms, Shor's algorithm is probabilistic—t*he correct answer is returned with a high probability, and the probability of failure decreases as the algorithm is repeated.* Shor's algorithm takes the following steps to find the prime factors of an integer N [1]:

1) We determine either N is an even number or a powers of prime number. If so, prime factorization is possible in a classical way.
2) We randomly choice a number x that satisfies $0 < x < N$.
3) We apply the greatest common divisor function $\gcd(x, N)$ on the given data and if
$$\gcd(x, N) \neq 1, \tag{1}$$
x is the prime factor of N.
4) In case equation (1) does not hold, we find the period r such that
$$f(a + r) = f(a) \tag{2}$$
where $f(a) = x^a \pmod N$.
5) We go back to the second step if either r is odd or r is even but
$$x^{r/2} + 1 = 0 \ \left(\bmod N\right). \tag{3}$$
6) In case r is even but equation (3) does not hold, the prime of factors of N are give as
$$\gcd\big(x^{r/2} \pm 1, N\big). \tag{4}$$

Here is it important to note that not all above steps use quantum algorithms. Particularly, classical computer takes exponential time in step 4. Therefore, step 4 is reconstructed as a phase estimation problem on quantum computer, reducing the required time to polynomial time. After that, the r is obtained with a high probability using the quantum Fourier transform.

In the later half of this section, we briefly describe phase estimation and quantum Fourier transform used in the circuit.

A. Quantum Fourier Transform

Quantum Fourier transform (QFT) is unitary transformation of qubits. QFT and FFT are cousin relations. QFT on a set of N dimensional orthonormal basis $|j\rangle$ is defined as:

$$\sum_{j=0}^{N-1} \alpha_j |j\rangle \rightarrow \sum_{k=0}^{N-1} \tilde{\alpha}_k |k\rangle , \tag{5}$$

where

$$\tilde{\alpha}_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi ijk/N} \alpha_j, \tag{6}$$

and $\tilde{\alpha}_k$ is the discrete Fourier transform of the amplitude α_k. Since QFT is also reversible like FFT, it satisfies the following

$$|\tilde{\phi}\rangle = F|\phi\rangle , \; F^\dagger F = I, \tag{7}$$

where given QFT matrix F is defined as

$$F = \frac{1}{\sqrt{N}} \sum_{j,k=0}^{N-1} e^{2\pi ijk/N} |k\rangle \langle j| . \tag{8}$$

ϕ can be expressed in binary, the relationship between $|\tilde{\phi}\rangle$ and $|\phi\rangle$ can be expressed as follows:

$$\begin{aligned} \phi &= \phi_1\phi_2 \cdots \phi_n, \\ &= \phi_1 2^{n-1} + \phi_2 2^{n-2} + \cdots + \phi_n, \end{aligned} \tag{9}$$

$$\begin{aligned} 0.\phi_l\phi_{l+1} \cdots \phi_m =&\phi_l/2 + \phi_{l+1}/4 + \cdots \\ &+ \phi_m/2^{m-l+1}, \end{aligned} \tag{10}$$

and

$$\begin{aligned} |\phi\rangle =& |\phi_1\phi_2 \cdots \phi_n\rangle \\ \rightarrow& \frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{2\pi i 0.\phi_n} |1\rangle\right) \\ &\left(|0\rangle + e^{2\pi i 0.\phi_{n-1}\phi_n} |1\rangle\right) \cdots \\ &\left(|0\rangle + e^{2\pi i 0.\phi_1\phi_2\cdots\phi_n} |1\rangle\right) , \end{aligned} \tag{11}$$

where n is the number of qubits in first register. The equation (11) can be achieved using Hadamard gate and controlled-R gate R_k [6] where R_k is defined as

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix} . \tag{12}$$

At the output, we perform measurements to determine the coefficients of the computational basis.

B. Phase Estimation

Phase estimation is used to find the eigenvalue ϕ, for a given unitary matrix U and eigenvector $|u_s\rangle$ where U is

$$U|y\rangle = |xy \bmod N\rangle . \tag{13}$$

We do not know eigenvector of U, but we know superposition of all the eigenvector, $|1\rangle$ [6]. Therefore, using concept of quantum parallelism, we can compute the phase of all eigenvectors in parallel. The input state passes through the controlled-U gate as follows:

$$\begin{aligned} \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle |1\rangle \rightarrow& \frac{1}{\sqrt{2^n}} \big(|0\rangle + e^{2\pi i(2^{n-1}\phi)} |1\rangle\big) \\ &\big(|0\rangle + e^{2\pi i(2^{n-2}\phi)} |1\rangle\big) \cdots \\ &\big(|0\rangle + e^{2\pi i(2^{0}\phi)} |1\rangle\big) |1\rangle , \end{aligned} \tag{14}$$

where ϕ is phase and n is the number of qubits in first register. If the real value ϕ is expressed in binary, it can be expressed as $\phi = 0.\phi_1\phi_2 \cdots \phi_n$. In the first register, the integer value resulting from 2^m (where m is integer which satisfied $0 < m < n$) can be ignored by multiplication of 2π. Then result of phase estimation is same with result of QFT. Therefore, after applying inverse QFT on first register, the superposition becomes

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |s/r\rangle |u_s\rangle , \tag{15}$$

where $|u_s\rangle$ is eigenvector. After measuring the first register, we obtained $|2^n\phi\rangle$ and get r with the continued fraction algorithm $\phi = s/r$ where s is the index corresponding to state $|2^n\phi\rangle$. This process uses classical processing.

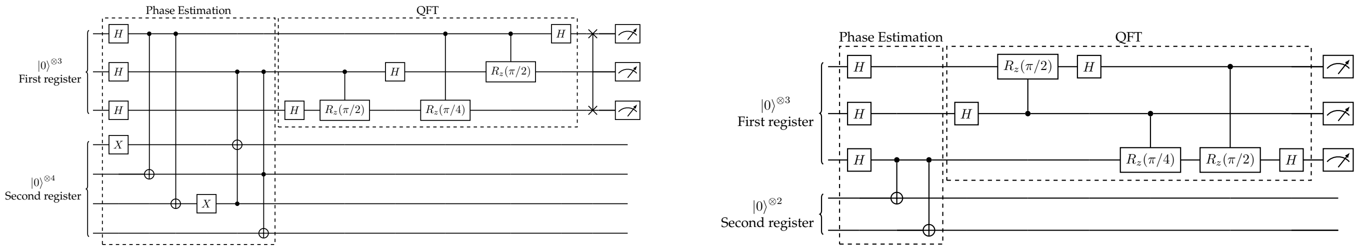
III. Compiled Circuit of Shor's algorithm

In this paper, we implement the circuit of prime factorization of $N = 15$ for $x = 7$, $x = 11$. Since large number of quantum gates are used to configure the phase, it is difficult to control error rate in phase estimation. Therefore, we implement Shor's algorithm as a compiled version of the circuit that simplified the phase estimation part. The circuit is designed with both the prime factors of N and r are known. In this section, we describe the compiled circuit for $N = 15$ when $x = 7$ and $x = 11$ [4], [7].

A. Finding period r, when N = 15, x = 7

Fig. 2(a) shows the schematic of our circuit for $N = 15$ and $x = 7$ where the first part is a simplified circuit of phase estimation, and the second part is an inverse QFT. The first register contained all the states of exponent k of the power of 7 with superposition, and the second register contained the result of 7^l mod 15, where $k = 2^n$ and l satisfies $0 \leq l < k$.

After applying inverse QFT in noiseless case, we apply the measurements to determine the phase ϕ. Since the measurement results are $|2^n\phi\rangle$ and the post-measurement states $|000\rangle , |010\rangle , |100\rangle$, and $|110\rangle$ occur with same probability we



(a) $N = 15$, $x = 7$

(b) $N = 15$, $x = 11$

Fig. 2. The compiled circuit of Shor's algorithm.

calculate ϕ corresponding to each post-measurement state as follow

$$2^3\phi = 000_2 = 0 \ \rightarrow \phi = \frac{0}{8} = 0, \quad (16)$$

$$2^3\phi = 010_2 = 2 \ \rightarrow \phi = \frac{2}{8} = \frac{1}{4}, \quad (17)$$

$$2^3\phi = 100_2 = 4 \ \rightarrow \phi = \frac{4}{8} = \frac{1}{2}, \quad (18)$$

$$2^3\phi = 110_2 = 6 \ \rightarrow \phi = \frac{6}{8} = \frac{3}{4}. \quad (19)$$

As $\phi = s/r$, we obtain $r \in \{2, 4\}$. To find the prime factors of N, it is necessary condition that r does not hold equation (3). In our case $r = 4$ satisfies the necessary condition and finally we get factors of 15 using equation (4).

B. Finding period r, when N = 15, x = 11

Similar to Fig. 2(a), we design our circuit for $x = 11$ as shown in Fig. 2(b). Here, the first register contains all the states of exponent k of the power of 11 with superposition, and the second qubit contained the result of 11^l mod 15. Following same procedure of $x = 7$, we perform the measurements after inverse QFT. The only difference is here the post-measurement states are $|000\rangle$, and $|100\rangle$ only and obtain ϕ from (16) and (18) followed by determining $r = 2$, respectively. Finally, we get factors of 15 using $\gcd(12, 15) = 3$ and $\gcd(10, 15) = 5$.

IV. Shor's algorithm on IBM device and MATLAB App Designer

In this paper, the above compiled version circuit is simulated through the MATLAB App Designer, and the simulation results were verified on the IBM device. To implement real quantum computer noise, the MATLAB App Designer applied two noise channel models: depolarizing channel and amplitude damping channel [8].

A. Gate modeling

Since depolarizing channel is an isotropic noise model, it produces uniformly distributed errors. The depolarizing channel transforms the input state ρ to the maximally mixed state π through parameter p and is specified by the following equation.

$$\mathcal{N}_{\mathrm{D}}(\rho) = (1 - p)\,\rho + p\pi, \quad (20)$$

TABLE I
The noise parameter of gates

	X	CNOT	U_1	Rz	H
p	0.162	0.0653	0.0594	0.22	0.0578
η	-	-	-	-	0.0024

Amplitude damping channel is a channel that embodies the noise that occurs when the spin state loses energy at one location and changes to another location. That is, noise generated when $|1\rangle$ changes to $|0\rangle$ is implemented. The action of the channel on the two-qubit input ρ is

$$\mathcal{N}_{\mathrm{A}}(\rho) = \sum_{j=0}^{1}\sum_{i=0}^{1} (K_i \otimes K_j)\,\rho\,(K_i \otimes K_j)^\dagger\,, \quad (21)$$

where the Kraus operator K_0 and K_1 for the one-qubit amplitude damping channel with damping parameter η are defined as

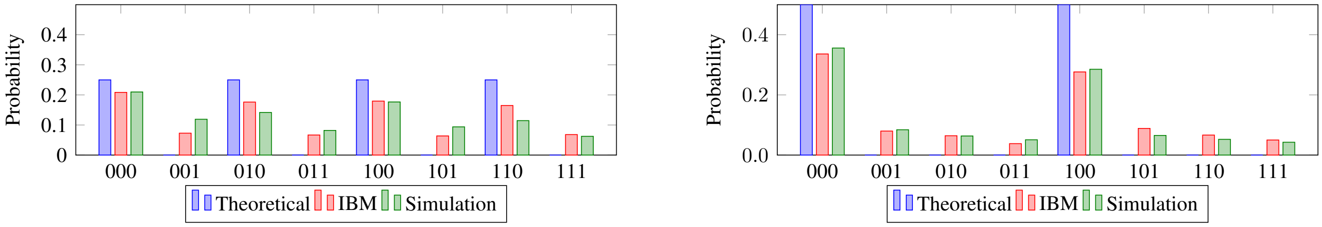
$$K_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\eta} \end{bmatrix}, \quad K_1 = \begin{bmatrix} 0 & \sqrt{\eta} \\ 0 & 0 \end{bmatrix}. \quad (22)$$

B. Simulation and Validation

We implement complied version of the Shor's algorithm in MATLAB App Designer, and validate our resutls by IBM device, Melbourne. Table. I shows the gate noise parameters to implement our circuit for Shor's algorithm on MATLAB App Designer where U_1 shows the unitary gate to decompose the Toffoli gate [9]. Each parameter is obtained based on results of our circuit implement on IBM device to validate our simulations. Fig. 3 shows the probabilities of post-measurement states of the implementation of compiled version of the Shor's algorithm on MATLAB App Designer without gate noise (purple bars), IBM quantum devices (pink bars) and MATLAB App Designer with gate errors (green bars). As Toffoli gate is decomposed into Hadamard gate H, CNOT gate and U_1 to implement complied version of Shor's algorithm, it increase the error probability in MATLAB App Designer simulation results as compared to IBM quantum devices results (see pink and green bars in Fig. 3).

V. Conclusion

In this paper, we simulated compiled version circuit of the Shor's algorithm on MATLAB App Designer and verified our



(a) $N = 15$, $x = 7$

(b) $N = 15$, $x = 11$

Fig. 3. Measurement statistics from the theory, IBM quantum device and MATLAB App Designer simulation.

results through an IBM device. We have modelled the behavior of IBM quantum gates as ideal gates on the noisy channel and benchmark the noisy behavior of IBM device on MATLAB App Designer. Although it is a compiled circuit, it includes all the elements necessary to implement a general version of the Shor's algorithm. The simulation result on MATLAB App Designer has similar trend with IBM quantum device as shown in Fig. 3. Currently, implementing the Shor's algorithm for the general case is actively researched, but at the current noise level, it quickly reaches the limit of the number of gates that can obtain a reasonable result. Implementing the Shor's algorithms through compiled circuit is a stepping stone towards the implementation of general circuit. The future work can be done to implement general circuit for any value of N.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C2007037).

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[2] X. Zhou and X. Tang, "Research and implementation of RSA algorithm for encryption and decryption," in *Proc. of 2011 6th international forum on strategic technology*, vol. 2. IEEE, 2011, pp. 1118–1121.

[3] C. Gidney and M. Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," *arXiv preprint arXiv:1905.09749*, May 2019.

[4] L. M. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance," *Nature*, vol. 414, no. 6866, pp. 883–887, 2001.

[5] E. Martin-Lopez, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O'brien, "Experimental realization of Shor's quantum factoring algorithm using qubit recycling," *Nat. Photonics*, vol. 6, no. 11, pp. 773–776, 2012.

[6] M. A. Nielsen and I. L. Chuang, "Quantum computation and quantum information," 2002.

[7] P. J. Coles, S. Eidenbenz, S. Pakin, A. Adedoyin, J. Ambrosiano, P. Anisimov, W. Casper, G. Chennupati, C. Coffrin, H. Djidjev *et al.*, "Quantum algorithm implementations for beginners," *arXiv preprint arXiv:1804.03719.*, 2018.

[8] S. Yun, K. Kwon, J. ur Rehman, F. Zaman, and H. Shin, "Quantum duplex coding for classical information on IBM quantum devices," in *Proc. of Korean Institute OF Communications and information Sciences (KICS) Autumn Conference*, Nov. 2019, pp. 415–418.

[9] V. V. Shende and I. L. Markov, "On the cnot-cost of toffoli gates," *Quantum Inf. Comput.*, vol. 9, pp. 461–486, 2009.